Getting Started on Pink

Harvey J. Wasserman, CCN-7

December 21, 2004

Topics

- Overview: Hardware & Software Architecture
  - What makes Pink different from other LANL systems.
  - Why we are making it different.
- Porting Considerations
- Compiling
- How to Run Jobs
- Filesystems

Aviso: all of this is new (to me, too) and evolving.

- Lecture assumes familiarity with LANL computing environment.

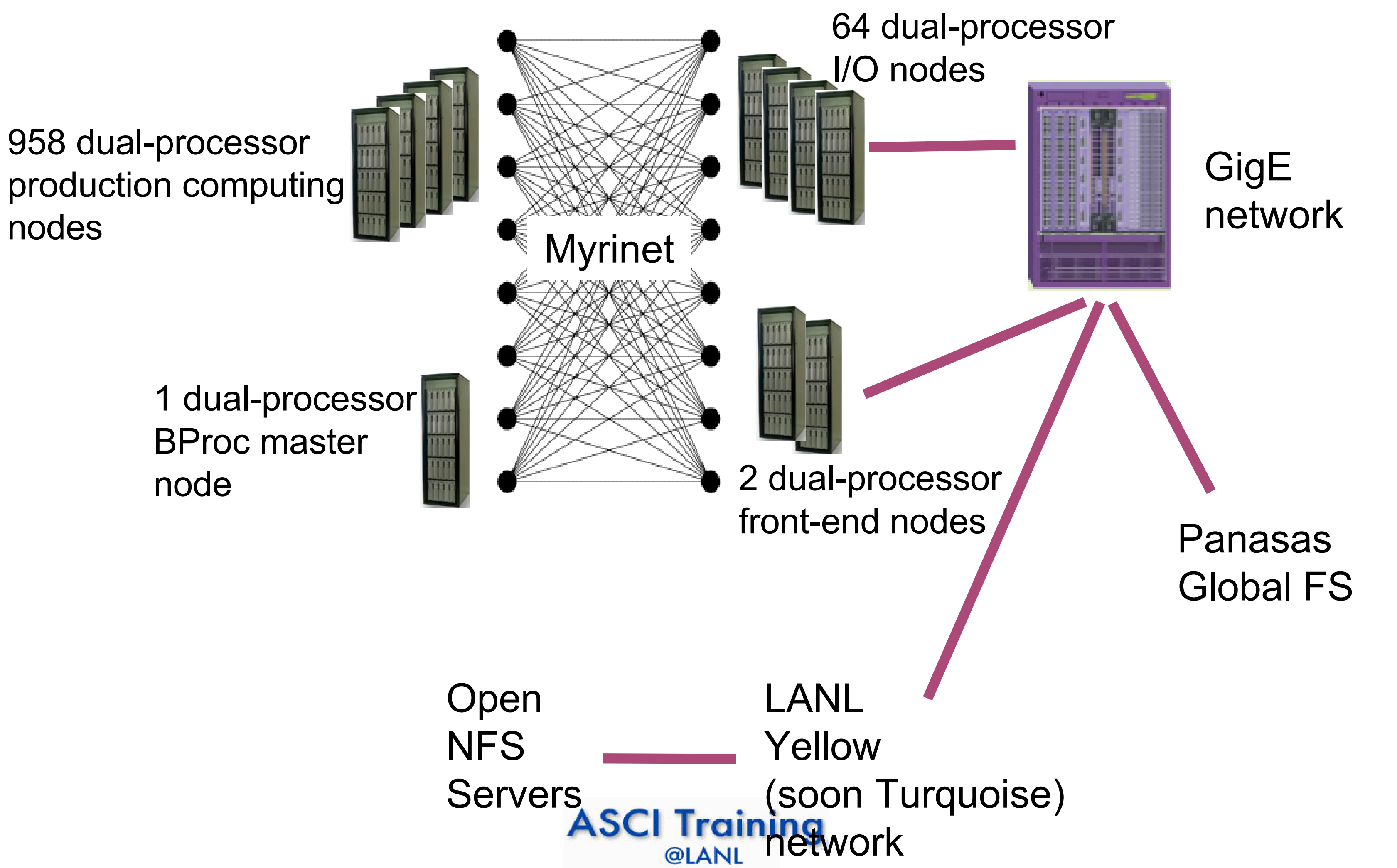
Pink Has Two Purposes

- Open production system for Institutional Computing workloads.
- A gateway to future computing strategies at LANL, supporting the following objectives:
  - Better price-performance platforms: Intel Xeon & Myrinet
  - Production-quality, open-source software: RH Linux
  - Vendor-independent HPC features: Linux NetworX, Panasas, CCS-1
  - High-availability cluster computing environment: ClusterMatic Science Applience

Pink Configuration

- 1,024 2-processor nodes
  - some front-end nodes & 64 fileserver nodes.
- Each node has:
  - 2 Intel 2.4-GHz Xeon Processors / 2 GB Memory
  - zero disk drives, zero ethernet cables ***
    - *** except for front ends
  - DDR SDRAM (instead of RDRAM) & 400-MHz system bus
  - Peak memory BW 3.2 GB/s; 1.6 GB/s more typical (compare to 2.1 GB/s on Alpha 21264)
- 9.8-TeraOps peak system performance.

Pink Configuration


958 dual-processor production computing nodes
1 dual-processor BProc master node
Myrinet
64 dual-processor I/O nodes
2 dual-processor front-end nodes
GigE network
Panasas Global FS
Open NFS Servers
LANL Yellow (soon Turquoise) network

Intel Xeon Microprocessor

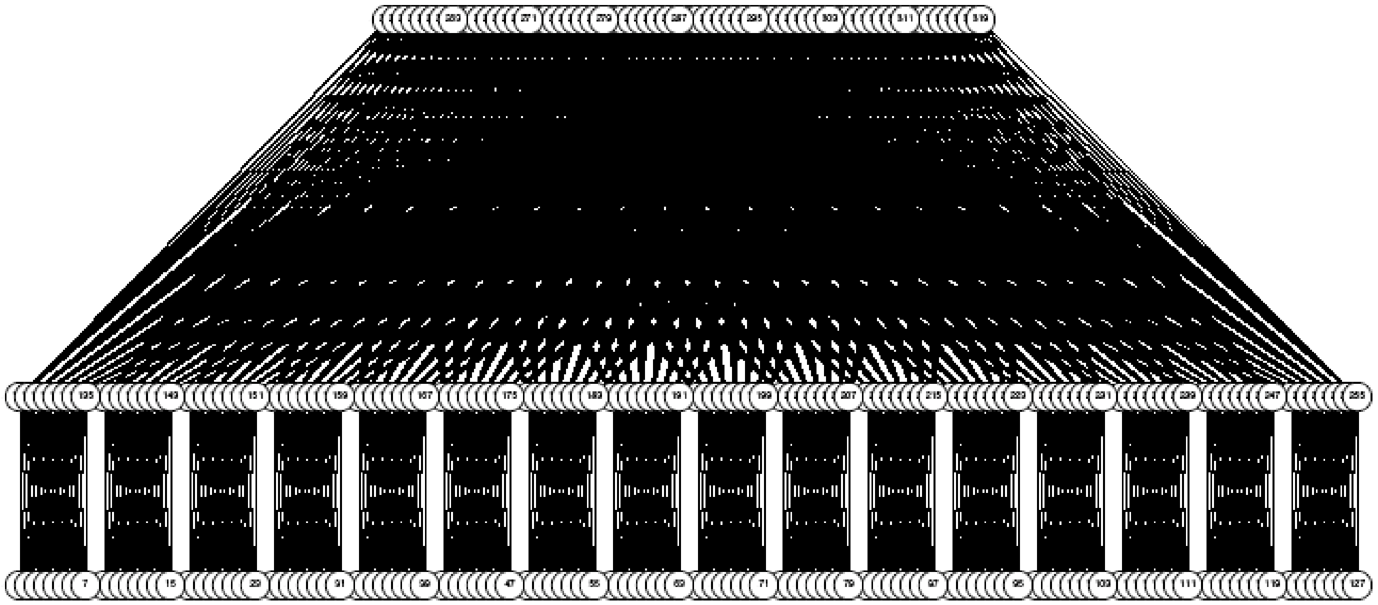
- **32-bit x86 (IA32) architecture.**
- **Xeon is basically a multiprocessor version of Intel Pentium IV**
- **The x87 FPU uses a floating point stack with eight 80-bit register elements. These same 80-bit registers are used whether the operands are single, double, or extended-double precision.**
- **SIMD "vector" units: MMX & SSE 2**
-
```
pink.lanl.gov-> more /proc/cpuinfo
model name      : Intel(R) Xeon(TM) CPU 2.40GHz
cache size      : 512 KB
```

Myrinet Interconnect

- Network interface card connects to nodes' PCI I/O bus + series of federated 16-port cross-bar switches + optical cables + Myrinet gm software + CCS-1 mapper
- There is one "rail" of interconnect.
- Myrinet should have a unidirectional MPI send/receive latency of 6-7 microseconds and a peak unidirectional transfer bandwidth of 250 MB/s. See CCS-3 results for details.

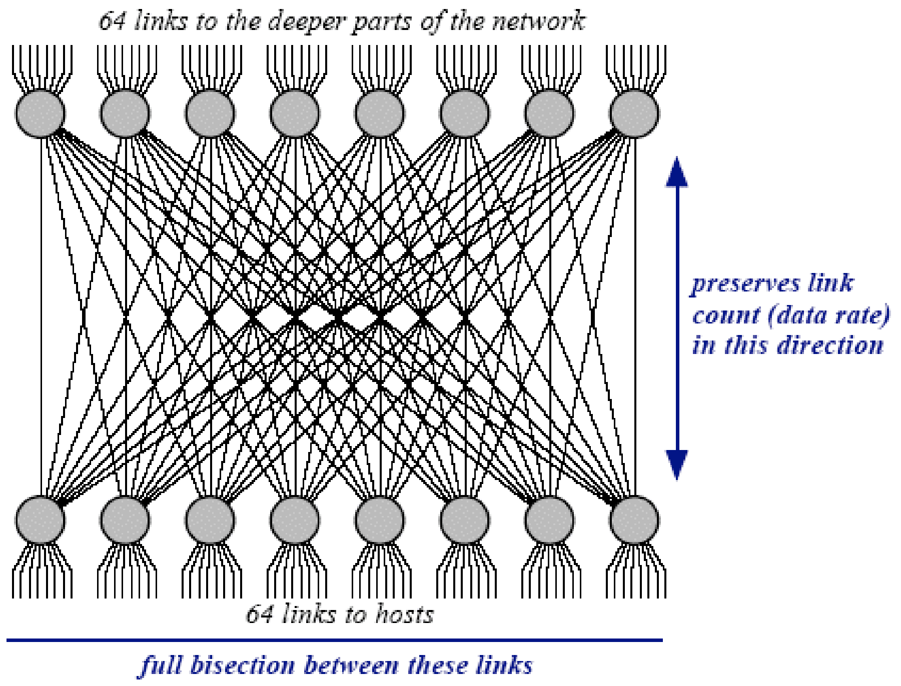
Myrinet Interconnect

1024 Hosts, 320 SW16 Switches



- 5-hop worse-case transfer
- Bandwidth-preserving topology (similar to fat-tree)

Myrinet Interconnect


64 links to the deeper parts of the network
preserves link count (data rate) in this direction
64 links to hosts
full bisection between these links

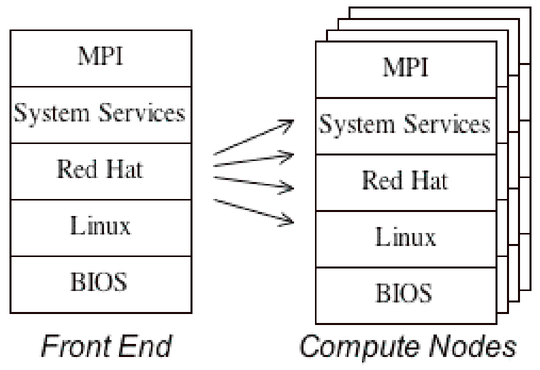
Pink Software Architecture

- Pink is an example of a "Science Appliance."
- Award-winning concept invented by LANL's CCS-1 Cluster Computing Team (Ron Minnich, TL).
- *Objective is to provide more computing cycles to users by making the cluster easier to build and manage.*
- Pink is essentially the world's largest Science Appliance. *
  - * Lightning has more nodes but is operated as 6 separate segments.

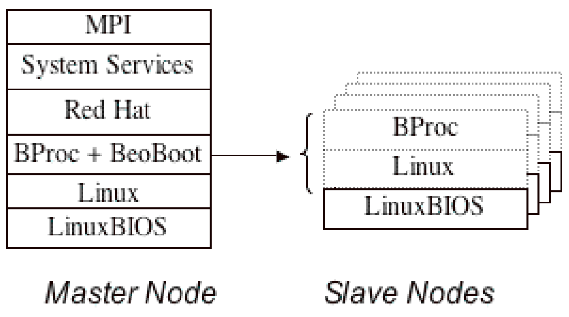
What is a Science Appliance?

- Science Appliance actually refers to a redesign of both hardware and software for large-scale clusters.
- The key software in a Science Appliance is a suite that LANL developed called "Clustermatic."
  - Clustermatic can completely control a cluster, from the BIOS up to a high level programming environment.
  - It features the Beowulf Distributed Process Space (**BProc**), LinuxBios, and a variety of other open-source kernel modifications, utilities, and libraries.
- Reliability improved by reducing HW and SW complexity; availability improved by reducing boot time to seconds or a few minutes.

Science Appliance vs. a Traditional Cluster



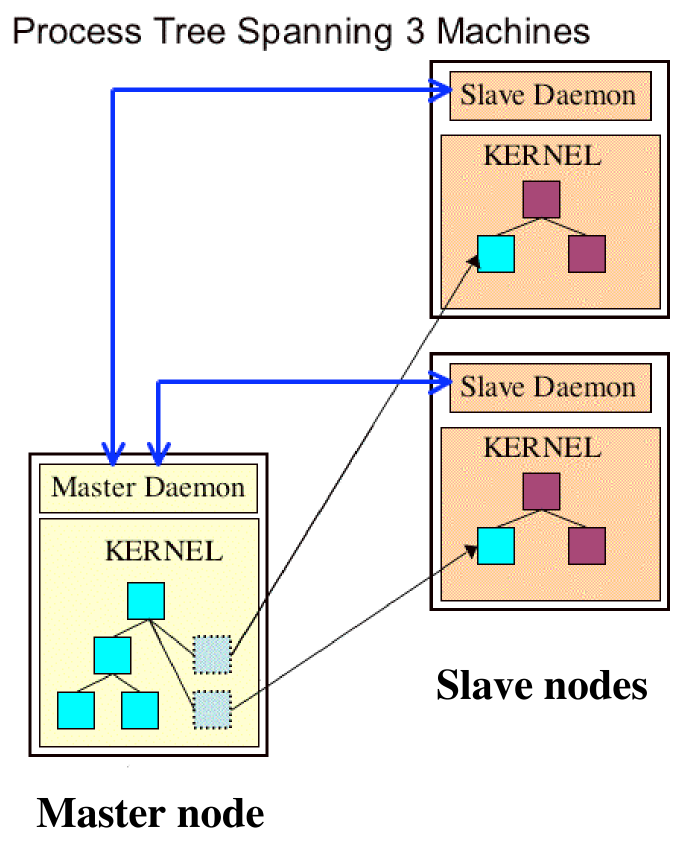
Traditional Cluster Architecture



Science Appliance Architecture

- A traditional cluster is built by replicating a complete workstation's software environment on every node.
- In a Science Appliance, we have <u>master nodes</u> and <u>slave nodes</u> but only the master nodes have a fully-configured system.
- The slave nodes run a minimal software stack consisting of LinuxBIOS, Linux, and BProc.
- No Unix shells running on the slave nodes, no user logins on the slave nodes.

Science Appliance vs. a Traditional Cluster



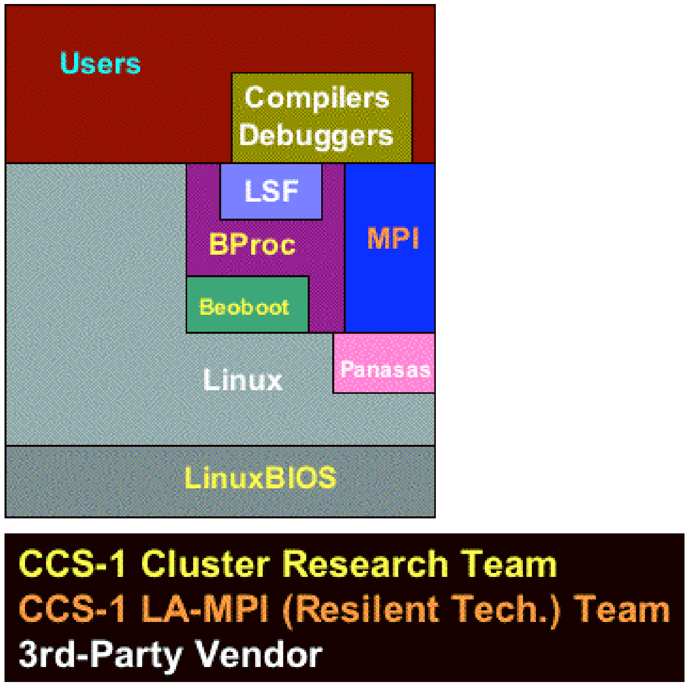
- Most importantly, BProc enables a distributed process space across nodes within the Pink cluster: all user processes running on the slave nodes appear as processes running on the front end.

- Users create processes on the master node and the system migrates them (the processes) to the slave nodes.

- Standard input, output, and error streams are redirected to the master node.

Processes remain visible, controllable on master.

Science Appliance Systems at LANL



- Lightning, Pink, Grendels, Flash
- MPI & LSF are BProc-integrated.
- File I/O environment is different among the various LANL BProc systems
- Result: LANL Science Appliance systems are easy to use but are different than other LANL systems (hence this presentation).
- Important note: The root filesystem in Clustermatic is RAM based!

Porting Considerations

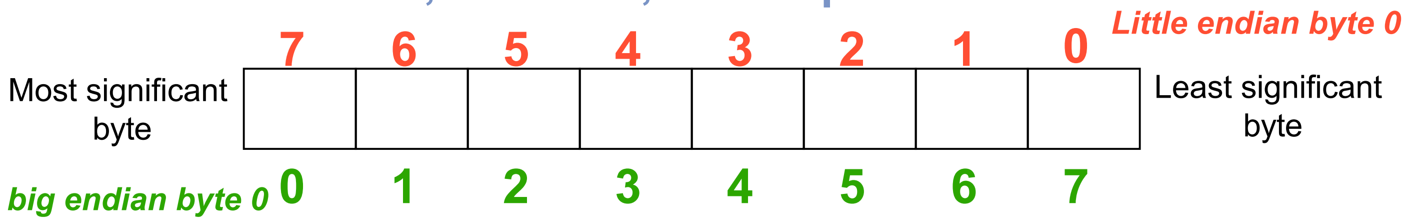
- Endianness
- Data Sizes
- 32-bit address space: 2-GB memory
- Large file support: Nominal limit is 2-GB files.
  - See "Big File Fix" .+3 view graphs

Porting Considerations: Endianness

Machine	Processor	Byte order
ASCI Q	DEC/Compaq/HP Alpha EV68	Little Endian
Lambda, Pink, Grendels	Intel Pentium	Little Endian
Blue Mountain or Theta	MIPS R10000	Big Endian
Mauve	Intel Itanium	Little Endian
Lightning	AMD Opteron	Little Endian

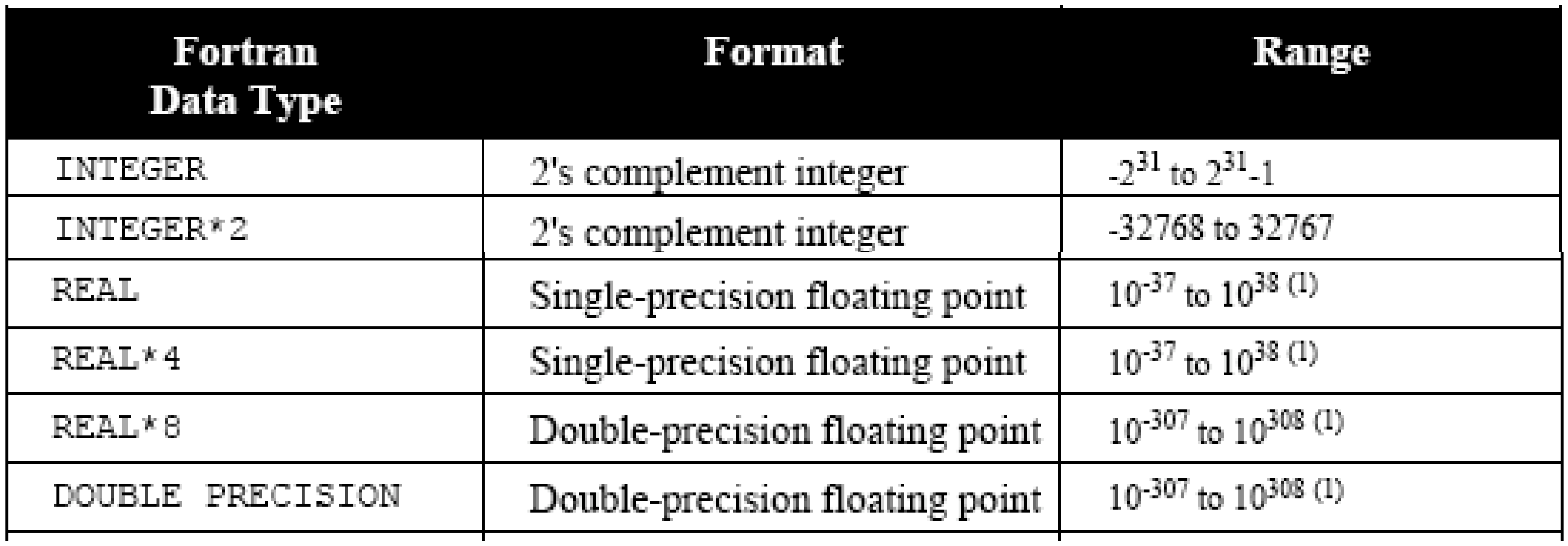
Porting Considerations: Endianness

- Big Endian: **address of most significant byte = word address (big end of the word).**
  - 0xDEADBEEF = DE AD BE EF
  - IBM 360, Power; Motorola 68k; MIPS; SPARC; HP PA
- Little Endian: **address of least significant byte = word address (little end of the word)**
  - 0xDEADBEEF = EF BE AD DE
  - Intel 80x86; DEC Vax; DEC Alpha



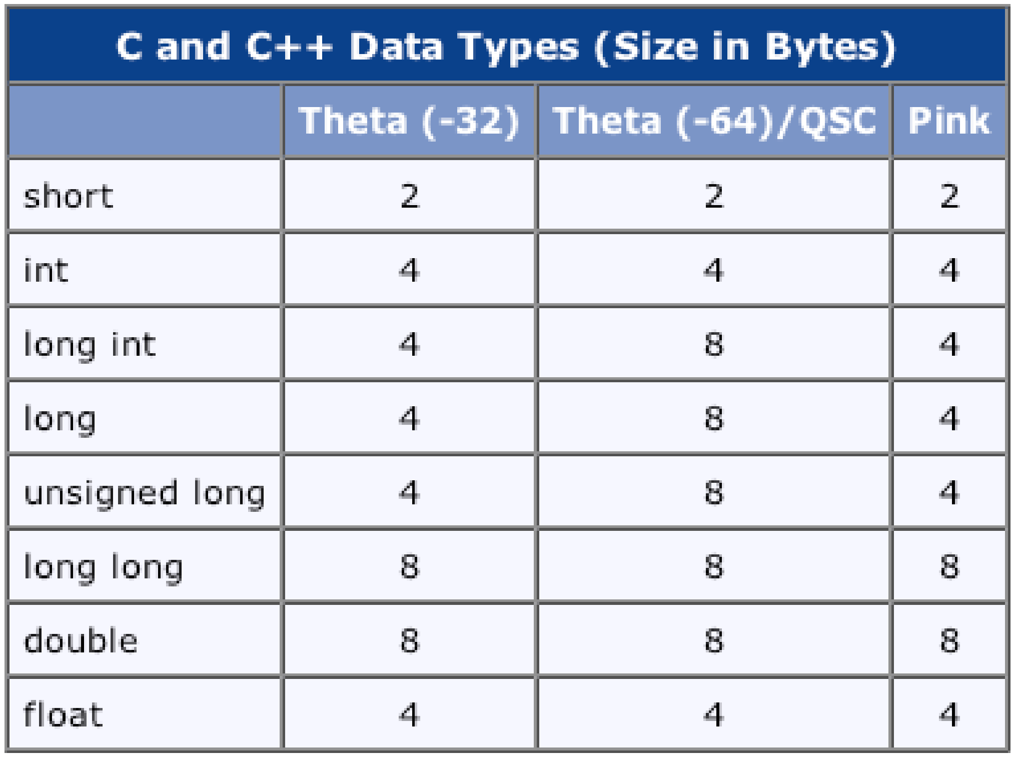
- **Code issues: pointer dereferencing can work differently**
- **Data file issues: Data files written on Theta must be converted to little-Endian format before they can be read on Pink.**
  - Some compilers have a compile-line option

Porting Considerations: Data Sizes



Fortran Data Type	Format	Range
INTEGER	2's complement integer	-2^{31} to $2^{31}-1$
INTEGER*2	2's complement integer	-32768 to 32767
REAL	Single-precision floating point	10^{-37} to 10^{38} (1)
REAL*4	Single-precision floating point	10^{-37} to 10^{38} (1)
REAL*8	Double-precision floating point	10^{-307} to 10^{308} (1)
DOUBLE PRECISION	Double-precision floating point	10^{-307} to 10^{308} (1)

Porting Considerations: Data Sizes



C and C++ Data Types (Size in Bytes)			
	Theta (-32)	Theta (-64)/QSC	Pink
short	2	2	2
int	4	4	4
long int	4	8	4
long	4	8	4
unsigned long	4	8	4
long long	8	8	8
double	8	8	8
float	4	4	4

Big File Fix

- The 32-bit address space limits files to 2 GB max.
- Linux kernel fixed this some time ago.
- Compile with
  ```
  -D_FILE_OFFSET_BITS=64  -D_LARGEFILE64_SOURCE=1
  D_LARGEFILE_SOURCE=1
  ```
- These preprocessor flags work with gcc and intel C as is.
- For PGI compilers add `-Mlfs`.
- Check that it worked with '`nm -B a.out`' and checking that things like '`open64`' are defined, not plain '`open`'.

Transferring Files

- HPSS via `k5psi` on pfe1, pfe2
- `psi` on pink, slave nodes
- `scp`
- However, plan is to move Pink to a new LANL network ("turquoise"); outside LANL firewall.

Modules

- The Modules package provides a convenient way for the system to make multiple versions of system software available and a convenient way for users to update their working environment.
- On Pink most system software - compilers, debuggers, parallel libraries - can be accessed *only* through the Modules package.
- The Modules package consists of two parts: a shell-level `module` utility/command and the modulefiles that the utility uses.
- On Pink you use the `module` utility on the front end.

Module Acts Differently on Pink

- Modulefiles have descriptive prefixes.
- You can use just the prefix in your commands to get the default version.

```
pink.lanl.gov-> module avail mpich
mpich/1.2.5(default) mpich/1.2.5-intel  mpich/1.2.5-pgi

pink.lanl.gov-> module load mpich

pink.lanl.gov-> module avail intel

intel/7.1(default) intel-c/8.1        intel-fortran/8.1

pink.lanl.gov-> module load intel

pink.lanl.gov-> module list
Currently Loaded Modulefiles:
  1) mpich/1.2.5   2) intel/7.1
```

Pink Modulefiles (12/14/2004)

- **Module avail**

```
intel/7.1(default)          pgi/5.2-4
intel-fortran/8.1           pgi/5.1(default)
intel-c/8.1
totalview/6.4.0-2(default)
lampi/1.5.6
lampi/1.5.7
lampi/1.5.8(default)
mpich/1.2.5(default)
mpich/1.2.5-pgi
mpich/1.2.5-intel
java2sdk/1.4.2_04
```

Compiling

- You will be compiling on the front-end system.
- Intel
  - Version 7.1 (default): one modulefile gets ifc (fortran compiler) and icc (C compiler)
  - Version 8.1: separate modulefile for ifort (fortran compiler) and icc
- Portland Group v5.1 and 5.2.4: `pgf77, pgf90, pgcc`
- Gnu: apparently don't need to load a modulefile

Intel Compilers

- `ifc` or `icc`, v7. `ifort`, v8.
- **`-convert big_endian -convert little_endian`**
- Optimization: `-O0`, {`O`, `-O1`, `-O2` are equivalent on IA32}, `-O3`
- **`-tpp7`**: Optimize build for Pentium 4 / Xeon.
- **`-Vaxlib:`** Fortran only - enables linking to compatibility library.
- **`-g:`** Build with debugging symbols. Intel lets you debug with optimization (`-O1` or `-O2`). `-O0` must be specified explicitly to turn all optimizations off.
- `-xW`: turns on auto-vectorizer (dubious value, IMHO)

Intel Compilers

- By default, all floating point exceptions (FPEs), such as invalid operation, denormal operand, overflow, underflow and divide-by-zero are masked on the IA32 architecture.
- Programs that encounter FPEs will not terminate unless steps are taken to unmask exceptions.
- No easy/direct way to catch FPEs with Intel `icc/ifc` v7. We are trying to import a method developed by LLNL to control FPEs. To use it, you will `#include` a header file `fpcontrol.h` and load with a library `libfpcontrol.a`. More information will be posted when available.

Compiling Miscellany

- CPP is in `/usr/bin/cpp`
- Perl is in `/usr/bin/perl` (not where it is on Theta or QSC)
- Nice trick (supplied by Mike McKay, CCN-8): Instead of executing your perl scripts with a hardwired directive line (e.g., `#!/usr/bin/perl`), use the following two lines as the first two lines in your script:

```
eval 'exec perl -w -S $0 ${1+"$@"}'
if 0;<rest of perl script>
```

MPI on Pink

- Two packages: LAMPI and MPICH
- MUST Load a modulefile, to compile and/or run
- Supports debugging with Totalview
- Must tell the compilers where to find include files and libraries if using LAMPI:

```
pfe1 -> module load lampi/1.5.6
pfe1 -> printenv |grep MPI
  MPI_ROOT=/usr/lampi-1.5.6/gm
  MPIHOME=/usr/lampi-1.5.6/gm
pfe1 -> pgf90 -I$MPI_ROOT/include         \
     -L$MPI_ROOT/lib -lmpi *.o
```

MPICH on Pink

- Do not use the regular compilers (ifc/pgf90). Use the MPICH compiler scripts, instead (mpif90, mpicc).
- The modulefile automatically adds these to your path.
- MUST Load a modulefile, to compile and/or run
- Supports debugging with Totalview
- Additional runtime command line arg REQUIRED: `--nper 2`
- No need to add libraries to the link line.
- No need to tell the compilers where to find include files and libraries:

```
pfe1-> module load  mpich/1.2.5-pgi
pfe1-> mpif90 *.o
pfe1-> mpirun -np 8 --nper 2 ./a.out
```

Additional MPI Note

- Modulefiles have dependencies, i.e., mpich/1.2.5-pgi depends on pgi/5.2-4 or pgi/5.1(default)
  - (Note difference in version #s)
- Have to load both the MPI and compiler modulefile in order to run.

```
pfe1 -> module load  mpich/1.2.5-pgi
   mpich/1.2.5-pgi(19):ERROR:151: Module
'mpich/1.2.5-pgi' depends on one of the module(s)
'pgi/5.2-4 pgi/5.1 '
     mpich/1.2.5-pgi(19):ERROR:102: Tcl command
execution   failed: prereq pgi
pfe1 -> module load  pgi mpich/1.2.5-pgi
```

Pink Filesystems

- Front end and back ends DO NOT have the same filesystems mounted.
- NFS Home directories (identical to lambda, QSC, etc.) mounted on front end ONLY.
- Front and back ends will have temporary workspace from Panasas filesystem:

  `/net/scratch1/userid` and `/net/scratch2/userid`
- Important: You want to do your runs in a directory that exists on both the front and back ends! Use the Pansas space.
  - May have to cd /net/scratch1/user_id ; mkdir user_id

Running Jobs on Pink

- Now the fun begins …

Logging In

- Like most other LANL production computing systems, the Pink cluster uses a front end to control access to the compute nodes.
- You log in to the front end and then you submit jobs, both interactive and batch, to the back-end compute servers.
- You do not log in to the back ends.
  - But you still may need to use `llogin`.
  - No login sessions on BProc slave nodes.

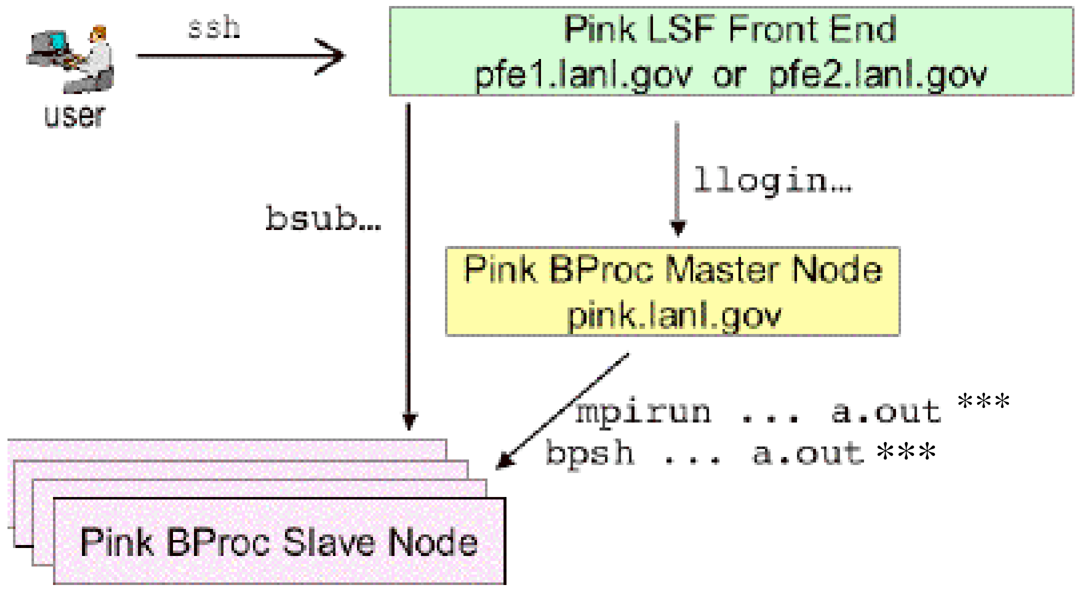
WARNING: CHANGES COMING

- Access to the current front-end system, pink.lanl.gov, will be removed on December 22.
- Front ends will be pfe1.lanl.gov and pfe2.lanl.gov
- pink.lanl.gov will become a BProc master node only - user access only through LSF.
- This is to reduce extraneous load on BProc caused by compiling, editing, file xfer, etc.

Running Jobs on Pink

- LSF is on Pink and you must use it to run jobs.
  - Both batch and batch-interactive jobs possible, as on other LANL systems.
- If you don't use LSF your job runs on the front end!
  - Different from other LANL systems.
- LSF commands are basically the same as on other LANL systems.
  - However, interactive use appears different.
- The job submission process potentially involves a combination of LSF and BProc commands.

Running Jobs on Pink



Procedure:

- Obtain an LSF allocation of slave nodes using `bsub` or `llogin`.
- Run job or script on master node.
- System migrates job to allocated slave nodes.

*** Illustration only; syntax will vary

Running Jobs on Pink

- Terminology:
  - "front ends"   pfe1 and pfe2
  - LSF "hosts"
    - "submit_hosts"   pfe1, pfe2, pink
    - "execution_hosts"   pink
  - BProc "master node" (pink) and "slave nodes."
- There is one LSF execution host, comprised of all the BProc slave nodes available for computing.
- Use `llogin` to reach the BProc master node.

Obtaining a Slave Node Allocation with LSF

- On Pink you can never have a shell on a back-end system.
- If you choose to run interactively (`llogin` or `BSUB -Is`) you will be allocated slave node processors by LSF but your shell will still be on the BProc master node.

```
pfe1.lanl.gov> llogin -n 8
Job <39681> is submitted to default queue <devq>.
<<Waiting for dispatch ...>>
<<Starting on pink>>

pink.lanl.gov> bjobs
JOBID    USER     STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
39681    hjw      RUN   devq       pfe1        8*pink      llogin     Dec 20 15:41
```

LSF on Pink

- Only the very basics considered here.
- If you don't know LSF, see
  http://computing.lanl.gov
  and documents referenced there.
- First we list the commands, then we talk about how to use LSF to get a slave node allocation.

LSF Commands: Host Status

- `lshosts` **shows static resource information for the machines, such as number of CPUs, total memory, total swap space, etc.**

```
pink.lanl.gov> lshosts
HOST_NAME        type    model   cpuf ncpus maxmem maxswp server RESOURCES
pink            BPROC   PC2400   46.0  1892  2021M      -    Yes ()
pfe1            BPROC   PC2400   46.0     -      -      -     No ()
pfe2            BPROC   PC2400   46.0     -      -      -     No ()
```

- `lsload` **gives dynamic load info**

LSF Commands: Host Status

- `bhosts` shows mixed dynamic/static information about batch processing on the LSF host.

Max job slots per user

Max job slots per host

job slots started

jobs running

```
pink.lanl.gov> bhosts -w
HOST_NAME            STATUS         JL/U    MAX  NJOBS    RUN  SSUSP  USUSP    RSV
pink                 ok                -   1892   1426   1418      0      0      8
```

Less than 1,916? Some nodes must be down

Other LSF Commands

- `bsub` **submits a job for batch or interactive execution**

  Usage: `bsub [options] a.out`

  `bsub [options] < bsub_scriptfile`
  - -n # processors. Only node-level alloc on Pink.
  - -W [hours:]minutes -q queue_name
  - -e error_filename -o output_filename
  - -Is submits a batch interactive job
- `llogin` **special version of bsub for interactive use.**
  - Puts you in $HOME with clean environment
- `bhist -a` **shows history of jobs, including ones that finished.**
- `bpeek` **shows** `stdout` **&** `stderr` **for unfinished batch jobs.**
- `bkill` **sends a job the** `SIGINT` **and** `SIGTERM` **signals.**

Other LSF Commands

- `bqueues` **displays information about LSF batch queues**

  Usage:
  ```
  bqueues
  bqueues -l
  bqueues -l queue_name
  bqueues -u user_id
  ```

```
pink.lanl.gov-> bqueues -u hjw
NAME     PRIO STATUS       MAX    JL/U JL/P JL/H NJOBS  PEND   RUN  SUSP
devq       7  Open:Active  -        32    1    -    24     0    24     0
batchq     5  Open:Active 1856    1024    1    -     0     0     0     0
```

- `bqueues -l batchq`

```
DEFAULT LIMITS:
RUNLIMIT
360.0 min of pink
MAXIMUM LIMITS:
RUNLIMIT                 PROCLIMIT
1440.0 min of pink       4 4 1024
```

Obtaining a Slave Node Allocation with LSF

- Note: Although your shell will still be on the front end after using `llogin`, it will be a *new* shell.
- Load your modulefiles after using `llogin` (`module load module_file_name`) .
- Recall that modulefile-related errors can be somewhat obscure. Example:

```
pfe1 -> module load lampi
pfe1 -> llogin
pink -> mpirun -np 8 sweep3d.mpi
 Not enough nodes to allocate all processes
pink -> module list
No Modulefiles Currently Loaded.
```

Obtaining a Slave Node Allocation with LSF

- LSF sets two important environment variables when it gives you a slave node allocation:
- `NODES` specifies which nodes the LSF job can use.
- `NODELIST` lists the processors on each node that the LSF job can use.

```
pfe1 -> llogin -n 6
Job <39681> is submitted to default queue <devq>.
<<Waiting for dispatch ...>>
<<Starting on pink>>

pink -> bjobs
JOBID   USER    STAT  QUEUE   FROM_HOST EXEC_HOST  JOB_NAME
39682   hjw     RUN   devq    pfe1      6*pink     llogin

pink -> env |grep NODE
NODES=4,5,11
NODELIST=4,4,5,5,11,11
```

Pink Queue Structure

	Priority	CPUs per Job Def / Max	Runlimit Def/Max hours	Queue Job Limit	User Job Limit	Notes
devq	7	2/8	12/12	-	16	ONLY_ INTERACTIVE
smallq	6	2 / 256	4/12	768	256	
largeq	5	258 / 1600	6/12	1032	1600	
nightq	5	- / 1600	2 / 10	-	1600	RUN_ WINDOW: 20:00-8:00

USER_SHARES: [institution, 35000] [support, 15000] [others, 5000]

Running Jobs on Pink

- Now you know how to get an allocation and what that looks like.
- What commands do you use to run jobs?
- The job submission process potentially involves a combination of LSF and BProc commands:

	Interactive	Batch

Sequential job	`llogin`
	`bpsh` *`node#`* `a.out`
		`bsub -n 2 'bpsh $NODES a.out'`

Parallel job	`llogin -`*`n`* `#`
	`mpirun -`*`np`* `# a.out`
		`bsub -`*`n`* `# mpirun -`*`np`* `# a.out`

Only 1 BProc Command You Need to Know

- `bpsh`
- Three others it couldn't hurt to know:

```
    bpstat     bpps     bptop
```

The BProc `bpsh` Command

- Pronunciation: bee-pish
- Runs a command on a slave node. Used for sequential commands. (Use `mpirun` for mpi.)
- Usage: bpsh [options] node# command [cmd args]
- Example: Run `ls -al /tmp/hjw` on node 3

```
bpsh 3 ls -al /tmp/hjw
```

- Common usage:

```
pink -> llogin
pink -> bpsh $NODES a.out < test_24.inp > test_24.out
```

- Caution: Does not execute a shell on back end

```
pink -> bpsh $NODES cd /tmp/hjw
bpsh: cd: command not found
```

Running MPI Jobs on Pink

- With LAMPI use

```
    mpirun -np # a.out.mpi
```

  to launch the job (plus LSF).
  - No BProc command needed.
- Extra manager process per node
- With MPICH use

```
    mpirun -np # --nper 2 a.out.mpich
```

- Caution: `mpirun` doesn't check for LSF allocation
  - Runs on front end if no allocation exists!
  - Can oversubscribe allocation without telling you.

Determining Job/System Status on Pink

- **Unix commands:** `ps, top`
- **LSF commands:** `bjobs`
- **BProc commands:** `bpstat, bpps, bptop`
- **Big difference:**
  - LSF command works on front ends and on BProc master node.
  - Unix commands and BProc commands work only BProc master node.

Determining Job/System Status on Pink

- `ps -ef` or `ps -f -u hjw`: slave node processes have their command names surrounded by [square brackets]

```
UID          PID  PPID  C STIME TTY          TIME CMD
hjw        30293 30093  1 09:13 ttyp7    00:00:00 mpirun -np 4
./sweep3d.mpi
hjw        30294 30293 19 09:13 ttyp7    00:00:00 [sweep3d.mpi]
hjw        30295 30293 19 09:13 ttyp7    00:00:00 [sweep3d.mpi]
hjw        30298 30295 87 09:13 ttyp7    00:00:01 [sweep3d.mpi]
hjw        30299 30294 92 09:13 ttyp7    00:00:01 [sweep3d.mpi]
hjw        30300 30294  0 09:13 ttyp7    00:00:00 [sweep3d.mpi]
hjw        30301 30295  0 09:13 ttyp7    00:00:00 [sweep3d.mpi]
```

- Watch for MPI processes without square brackets mistakenly running on front end!
- Unix `top` command will NOT show back end processes with square brackets.
- Can use Unix `kill` command on front end to send signal to back end processes.

Determining Job/System Status on Pink

- **LSF `bjobs` command**

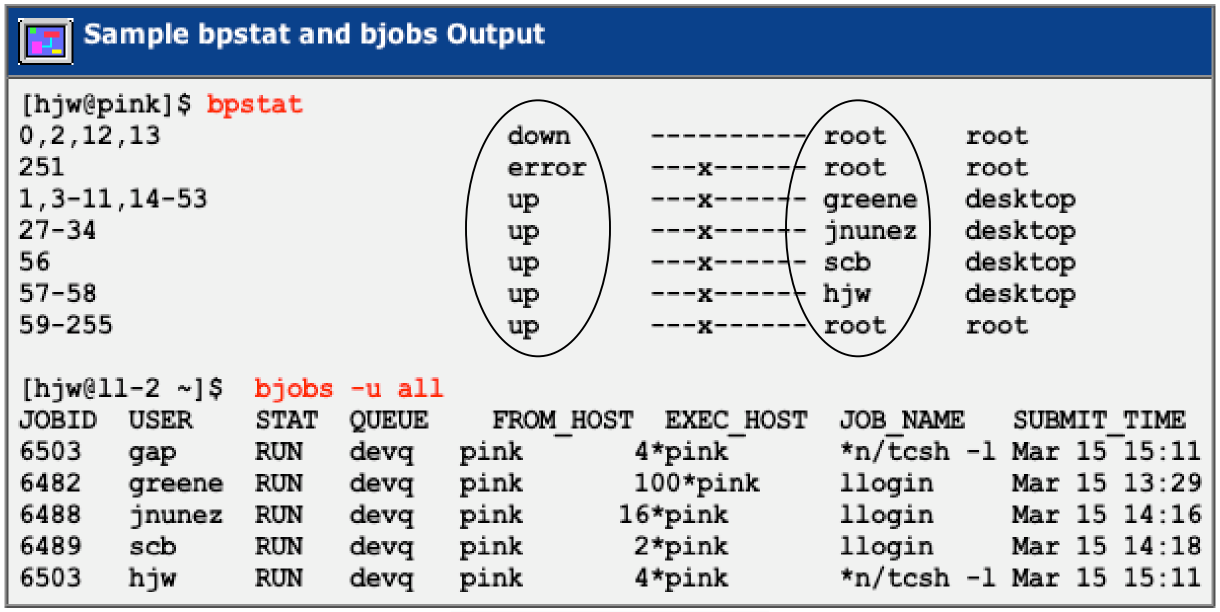
```
pfe1.lanl.gov->  bjobs

JOBID    USER    STAT  QUEUE   FROM_HOST   EXEC_HOST JOB_NAME SUBMIT_TIME
356      hjw     RUN   devq    pfe1        4*pink    llogin   Jun 17 09:04


pfe1.lanl.gov->  bjobs -u all

JOBID   USER     STAT  QUEUE FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
354     daughto  RUN   devq  pfe1        16*pink     island3    Jun 1621:45
356     hjw      RUN   devq  pfe1        4*pink      llogin     Jun 1709:04
```

- **Show jobs for a single user:** `bjobs -u hjw`
- **Show jobs that are pending and why:** `bjobs -lp`

The BProc `bpstat` Command

- **BProc `bpstat` command: Shows status of nodes**
  - `up` node is up and available
  - `down` node is down or can't be contacted by master
  - `boot` node is coming up (running node_up)
  - `error` an error occurred while the node was booting

Node #s



```
[hjw@pink]$ bpstat
0,2,12,13                       down    ----------  root     root
251                             error   ---x------  root     root
1,3-11,14-53                    up      ---x------  greene   desktop
27-34                           up      ---x------  jnunez   desktop
56                              up      ---x------  scb      desktop
57-58                           up      ---x------  hjw      desktop
59-255                          up      ---x------  root     root

[hjw@ll-2 ~]$  bjobs -u all
JOBID  USER    STAT  QUEUE    FROM_HOST  EXEC_HOST  JOB_NAME     SUBMIT_TIME
6503   gap     RUN   devq    pink         4*pink    *n/tcsh -l  Mar 15 15:11
6482   greene  RUN   devq    pink         100*pink  llogin      Mar 15 13:29
6488   jnunez  RUN   devq    pink        16*pink    llogin      Mar 15 14:16
6489   scb     RUN   devq    pink         2*pink    llogin      Mar 15 14:18
6503   hjw     RUN   devq    pink         4*pink    *n/tcsh -l  Mar 15 15:11
```

Determining Job/System Status on Pink

- Special CCN-7 version of `bpstat` command: `bpps`
- Only shows slave node activity.

```
pfe1 -> bpps
NODE  USER        PID  PGID S STIME      TIME COMMAND
   5  hjw       31009 31008 S 09:45 00:00:00 [sweep3d.mpi]
   6  hjw       31010 31008 S 09:45 00:00:00 [sweep3d.mpi]
   6  hjw       31013 31008 R 09:45 00:00:06 [sweep3d.mpi]
   5  hjw       31014 31008 R 09:45 00:00:06 [sweep3d.mpi]
   5  hjw       31015 31008 R 09:45 00:00:06 [sweep3d.mpi]
   6  hjw       31016 31008 R 09:45 00:00:06 [sweep3d.mpi]
   8  lpm       14822 14597 R 09:31 00:01:01    [mpihello]
   8  lpm       14823 14868 R 09:31 00:01:01    [mpihello]
```

- Returns nothing if no slave node processes

Common BProc Pitfalls

- The BProc commands DO NOT work on the front-end systems pfe1 and pfe2.

```
pfe1.lanl.gov> bpstat
bproc_nodelist: Input/output error

pfe1.lanl.gov> bpps
Bproc::proclist: Cannot allocate memory

pfe1.lanl.gov> bpsh 3 ./sweep3d.single
bproc_vexecmove_io: Function not
implemented
```

Common BProc Pitfalls

- `bsub -n 2 bpsh $NODES a.out` **FAILS**

Result: `NODES: Undefined variable.`

Correct behavior is obtained by quoting:

```
bsub -n 2 'bpsh $NODES hostname'
```

Common BProc Pitfalls

- `bpsh 6 ls /tmp; ls/tmp/hjw` **FAILS because the semicolon is a shell command separator.**

  Result: `ls /tmp` **runs on node 6**
  but `ls /tmp/hjw` **runs on the front end!**

- `bpsh $NODES ls > ~hjw/out` **Works, no problem, but immediately thereafer:** `bpsh $NODES ls ~hjw/out` **FAILS**
  Result:

  ```
  ls: /users/hjw/out: No such file or directory
  ```

- **Common theme: shell interpretation of a** `bpsh` **command takes place on the FRONT END only.**

Common BProc Pitfalls

```
[hjw@pfe1 ~/SWEEP/PINK]$ llogin -n 6
Job <39887> is submitted to default queue <devq>.
<<Waiting for dispatch ...>>
<<Starting on pink>>
[hjw@pink ~]$ cd SWEEP/PINK
[hjw@pink ~/SWEEP/PINK]$ module load lampi
[hjw@pink ~/SWEEP/PINK]$ mpirun -np 6 ./sweep3d.mpi
LA-MPI: *** mpirun (1.5.9)
LA-MPI:client/adminMessage.cc:930:
adminMessage::serverConnect timeout 42 exceeded --
0 client sockets account for 0 processes!
LA-MPI was unable to start your application.
This may be because:
<snip>
```

No Slave Node Directory

Common BProc Pitfalls

No Slave Node Allocation

```
pfe1-> llogin
Job <39882> is submitted to default queue <devq>.
<<Waiting for dispatch ...>>
<<Starting on pink>>
pink->  bpsh 3 ./sweep3d.single
3: Operation not permitted
pink-> bpstat
Node(s)                      Status  Mode        User      Group
157,278,701,740,778,889      down    ----------  root      root
0-156,158-277,279-513        up      ---x------  mswarren  desktop
514                          up      ---x------  esd       desktop
515                          up      ---x------  hxv       desktop
516-517                      up      ---x------  hjw       desktop
```

Debugging with Totalview

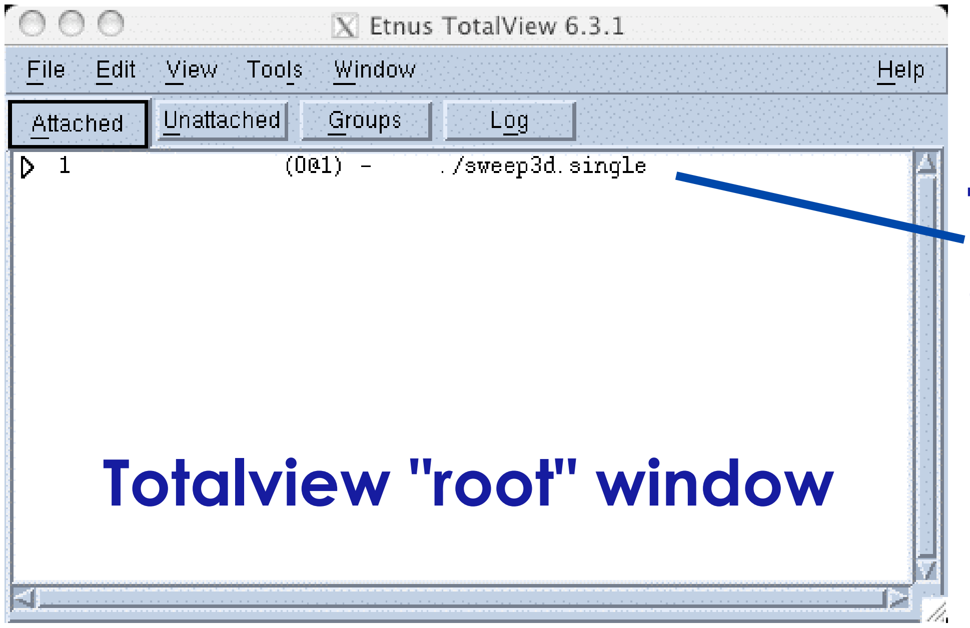
- Don't forget that both your source and executable must be visible on the slave nodes.

- The process on Pink differs for serial and parallel codes.

Debugging with Totalview on Pink

- Debug a serial code:

```
llogin
module load totalview/6.3.1
totalview -remote $NODES ./a.out
```


Etnus TotalView 6.3.1
File Edit View Tools Window Help
Attached Unattached Groups Log

```
▷ 1          (0@1) -     ./sweep3d.single
```

Totalview "root" window

To start debugging, double click on this line.

Debugging with Totalview on Pink

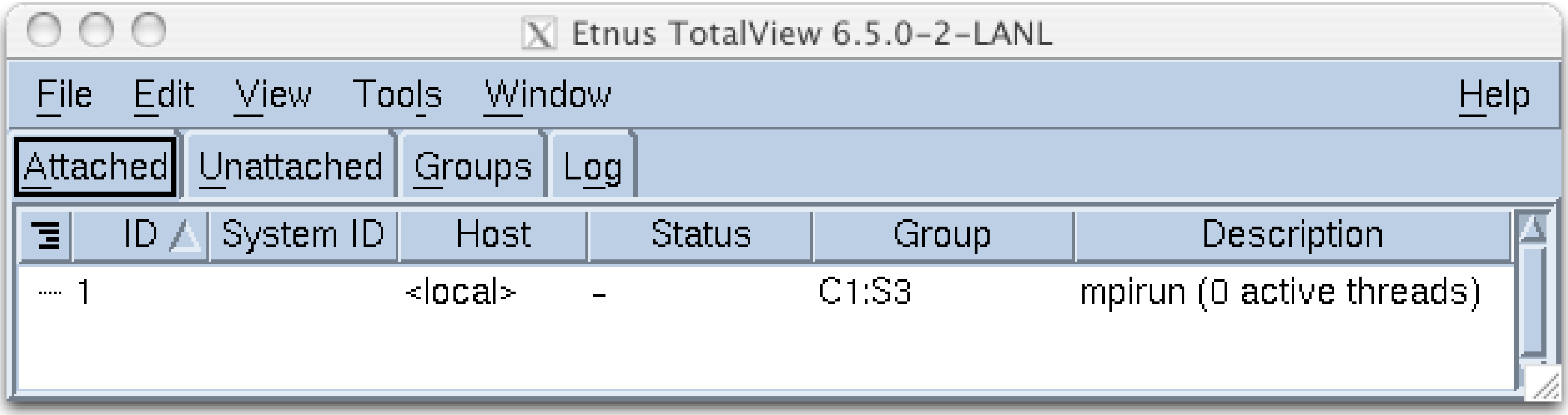
- Debug a parallel code:

```
llogin -n #
module load lampi totalview
totalview mpirun -a -np # ./a.out
```

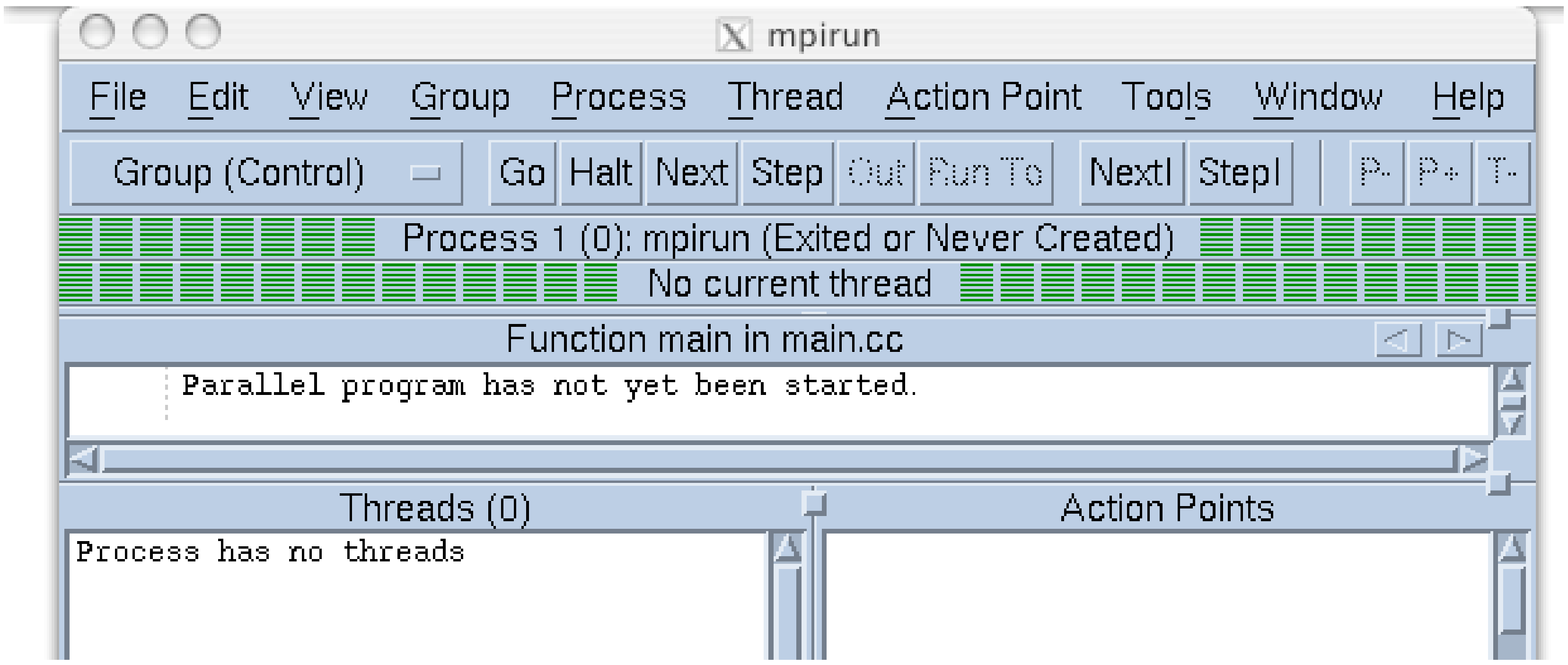
- Currently a minor complication with LA-MPI: have to link in the additional binary "`debug_gate.o`" which is in the same directory pointed to by `$MPI_ROOT/lib`.

- Debugging proceeds as it does on other LANL systems:

Totalview Root and Process Windows



ID	System ID	Host	Status	Group	Description
1		<local>	-	C1:S3	mpirun (0 active threads)



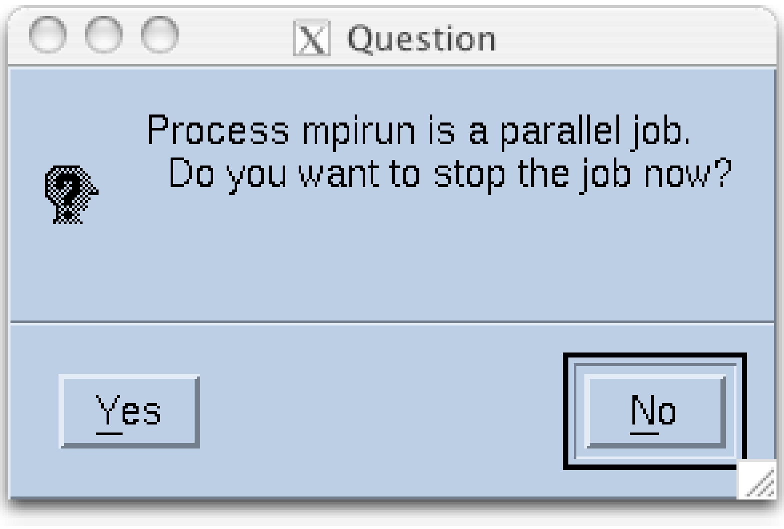
```
Process 1 (0): mpirun (Exited or Never Created)
No current thread
```

```
Parallel program has not yet been started.
```

```
Process has no threads
```

Totalview Parallel Debugging

- Answer "Yes" to set breakpoints, see source, etc.

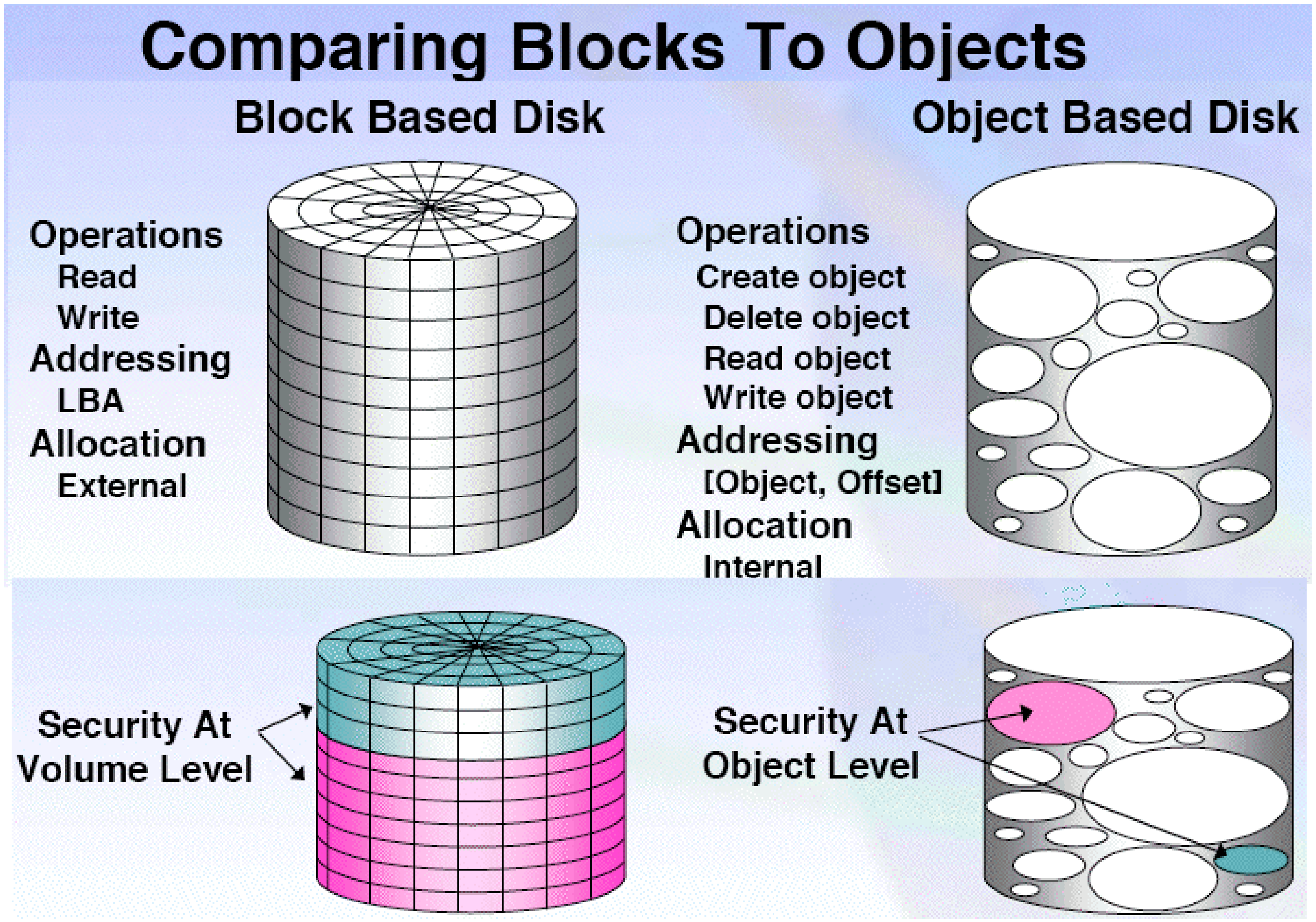


Process mpirun is a parallel job.
Do you want to stop the job now?

Panasas

- Object Based Storage: all I/O operations performed on arbitrarily-named data objects of variable size rather than sequentially numbered fixed-size blocks.
- The drives know about relationships among data objects. Reads and writes are directed to the object and an offset rather than to a logical block address.
- The Object Based Storage Devices stripe data across Secured storage devices. Rebalancing of the system is done automatically and is object based.
- To the user, the Panasas system should appear as a global shared file system with relatively high performance, even for small, sequential file I/O.
- Company founded by Garth Gibson. http://www.panasas.com

Panasas



Comparing Blocks To Objects

Block Based Disk

Operations
- Read
- Write

Addressing
- LBA

Allocation
- External

Object Based Disk

Operations
- Create object
- Delete object
- Read object
- Write object

Addressing
- [Object, Offset]

Allocation
- Internal

Panasas

- In the Panasas system 10 Storage Blades and 1 Director Blade (containing the metadata managers) are combined into a single shelf containing 5 TB of storage and a 16-port Gigabit Ethernet switch (which uses 4 ports to the network and 11 to the blades). The blades use Intel Pentium processors and 250-GB ATA disk drives.

11-blade shelf
H:7" [4U] x W: 19"

- On Pink, I/O goes over Myrinet to fileserver nodes, which then route I/O requests to Panasas. (Not that way on Lightning.) (Currently.)

Pink Summary

- `ssh pfe1.lanl.gov` **or** `ssh pfe2.lanl.gov`
- `llogin` **only if you need to run or query job status interactively (still have shell on front end)**
- `module load` **<compiler> <mpi>**
- `cd  /net/scratch1/userid.` **(Run from here.)**
- `llogin` **+** `mpirun -np # a.out.mpi` **or** `bsub -n # mpirun -np # a.out.mpi`
- `bpsh node# a.out.serial`
- **Make sure you don't execute a.out on front end!**
- `bpstat, bjobs, ps -ef, bpps`

Panasas User Best Practices

- See http://computing.lanl.gov/article/439
- Use `/usr/bin/panfs_df /net/scratch1` command instead of `df`.
- Small file performance is good, but highest bandwidths come from large I/O request sizes, larger than 100KB.•
- For parallel N-to-1 I/O with you may be able to improve bandwidth using MPI / IO hint `mpio_concurrent_write` assuming your files do not overlap (or have ghost cells on the overlapping edges).

Where To Go For Help

- ICN Consultants support Pink users.
- 5-4444 option 3 or e-mail: consult@lanl.gov
- Do NOT send e-mail to sysadmins.
- http://computing.lanl.gov
- http://asci-training.lanl.gov
- Pink status eventually on

  http://icnn.lanl.gov

  and

  http://icnn.lanl.gov/drm/alljobs

Want More Info?

- BProc Project Description http://bproc.sourceforge.net
- LANL's Cluster Research Team http://public.lanl.gov/cluster
- Clustermatic Home Page http://www.clustermatic.org
- LANL's Pink System http://www.lanl.gov/projects/Pin
- Linux Bios Home Page http://www.linuxbios.org
- A news article about Lightning from fcw.com
- Linux Networx's Home Page http://www.linuxnetworx.com
- Linux Networx's take on LinuxBios linuxnetworx.com/products/linuxbios.php
- Scyld Computing Corp., a company selling BProc systems
- Linux Labs

Acknowledgments

- This presentation is based on the combined work of many people, including
  - ICN Consultants (Rob Cunningham, Sara Hoshizaki, Jeff Johnson, David Kratzer, Hal Marshall, Roger Martz, and Meghan Quist).
  - Linux System Admin Team (Sean Blanchard, Jerry DeLapp, Daryl Grunau, Dave Neal).
  - CCS-1 Cluster Team (Sung-Eun Choi, Erik Hendriks, Ron Minnich, and others)
  - CCS-7 Scientific Computing Resources (SCR) Team, especially Stephany Bouchier and Ernie Buenafe
  - Blaise Barney (LLNL)
- Any errors in this presentation are due to hjw@lanl.gov only.